Sep 10, 1993 13:21 | GOPAL C. RAY | Chg_Scr

Interrupt | Escape | Rec_Def | Save Cs | NDC-Management | DocID-Mainte e | C/sc_list

Move | Patent Number Retrieval Parameter Menu | Close

Add new document identifiers to be added to this NDC

(1)	(2)	(3)	(4)
(5)	(6)	(7)	(8)
(9)	(10)	(11)	(12)
(13)	(14)	(15)	(16)
(17)	(18)	(19)	(20)
(21)	(22)	(23)	(24)
(25)	(26)	(27)	(28)
(29)	(30)	(31)	(32)
(33)	(34)	(35)	(36)
(37)	(38)	(39)	(40)
(41)	(42)	(43)	(44)
(45)	(46)	(47)	(48)
(49)	(50)	(51)	(52)
(53)	(54)	(55)	(56)
(57)	(58)	(59)	(60)
(61)	(62)	(63)	(64)
(65)	(66)	(67)	(68)
(69)	(70)	(71)	(72)
(73)	(74)	(75)	(76)
(77)	(78)	(79)	(80)
(81)	(82)	(83)	(84)
(85)	(86)	(87)	(88)
(89)	(90)	(91)	(92)
(93)	(94)	(95)	(96)
(97)	(98)	(99)	(100)

Named Document Collection: 917503.fir

Pick a display order

Pick an overall order: M
- (S) Separate subclasses
- (M) Merge subs together

Pick a date order: N
- (N) Newest-to-oldest
- (O) Oldest-to-newest

Pick a duplicates option: D
- (D) Do not show duplicates
- (S) Show duplicates

List patent sections to display in desired order: F, D

(F)	Front page	
(D)	Drawings	
(S)	Specification	
(S1)	First page of Specification	(US only)
(S2)	First two pages of Specification	(US only)
(C)	Claims	
(CC)	Changes/Corrections	(US only)
(R)	Reexamination certificates	(US only)
(AM)	Amendments	(Foreign only)
(A)	All sections in standard order	

Pick a viewing option: A
- (U) View unreviewed
- (S) View skipped
- (T) View tagged
- (A) View all
- (N) View none

Retrieve Documents from Training File? (Y / N): N

Execute? (Y / N): y

Sep 10, 1993 13:57 | GOPAL C. RAY | Chg_Scr

Interrupt | Escape | Rec_Def | Save Cs | NDC-Management | DocID-Mainte ce | C/sc_list

Move **Patent Number Retrieval Parameter Menu** Close

Add new document identifiers to be added to this NDC

(1)	(2)	(3)	(4)
(5)	(6)	(7)	(8)
(9)	(10)	(11)	(12)
(13)	(14)	(15)	(16)
(17)	(18)	(19)	(20)
(21)	(22)	(23)	(24)
(25)	(26)	(27)	(28)
(29)	(30)	(31)	(32)
(33)	(34)	(35)	(36)
(37)	(38)	(39)	(40)
(41)	(42)	(43)	(44)
(45)	(46)	(47)	(48)
(49)	(50)	(51)	(52)
(53)	(54)	(55)	(56)
(57)	(58)	(59)	(60)
(61)	(62)	(63)	(64)
(65)	(66)	(67)	(68)
(69)	(70)	(71)	(72)
(73)	(74)	(75)	(76)
(77)	(78)	(79)	(80)
(81)	(82)	(83)	(84)
(85)	(86)	(87)	(88)
(89)	(90)	(91)	(92)
(93)	(94)	(95)	(96)
(97)	(98)	(99)	(100)

Named Document Collection: 917503.fir

Pick a display order

Pick an overall order: M
- (S) Separate subclasses
- (M) Merge subs together

Pick a date order: N
- (N) Newest-to-oldest
- (O) Oldest-to-newest

Pick a duplicates option: D
- (D) Do not show duplicates
- (S) Show duplicates

List patent sections to display in desired order: F, D
- (F) Front page
- (D) Drawings
- (S) Specification
- (S1) First page of Specification (US only)
- (S2) First two pages of Specification (US only)
- (C) Claims
- (CC) Changes/Corrections (US only)
- (R) Reexamination certificates (US only)
- (AM) Amendments (Foreign only)
- (A) All sections in standard order

Pick a viewing option: U
- (U) View unreviewed
- (S) View skipped
- (T) View tagged
- (A) View all
- (N) View none

Retrieve Documents from Training File? (Y / N): N

Execute? (Y / N): y

Move | Text Search | Close

```
=> s (364*241.2/ccls or interrupt#)(p)hardware(p)software
          1164 364*241.2/CCLS
                 (364/241.2/CCLS)
         57017 INTERRUPT#
         43378 HARDWARE
         28842 SOFTWARE
L1        1002 (364*241.2/CCLS OR INTERRUPT#)(P)HARDWARE(P)SOFTWARE

=> s l1 (p) (condition# or failure# or error#)
              855562 CONDITION#
        130123 FAILURE#
        147714 ERROR#
      L2           285 L1 (P) (CONDITION# OR FAILURE# OR ERROR#)

=>
```

INPUT:

Sep 10, 1993 13:40 | GOPAL C. RAY | Chg_Scr

Interrupt | Hold/Res | Clr_Out | Inp f | NDC_Add | Pg/Scr_Mode | Prt_All | P Rem | Cont_Prt | Add_Blk | Prt_Blk

Move | Text Search | Close

```
=> s (364*941/ccls or interrupt#)(p)hardware(p)software
           573 364*941/CCLS
                 (364/941/CCLS)
         57017 INTERRUPT#
         43378 HARDWARE
         28842 SOFTWARE
L1        1002 (364*941/CCLS OR INTERRUPT#)(P)HARDWARE(P)SOFTWARE

=> s l1 (p) (condition# or failure# or error#)
            855562 CONDITION#
        130123 FAILURE#
        147714 ERROR#
      L2        285 L1 (P) (CONDITION# OR FAILURE# OR ERROR#)

=>
```

INPUT:

Move Text Search Close

```
=> s interrupt#(p)hardware(p)software
        57017 INTERRUPT#
        43378 HARDWARE
        28842 SOFTWARE
L1       1002 INTERRUPT#(P)HARDWARE(P)SOFTWARE

=> s l1 (p) (condition# or failure# or error#)
            855562 CONDITION#
       130123 FAILURE#
       147714 ERROR#
      L2          285 L1 (P) (CONDITION# OR FAILURE# OR ERROR#)

=>
```

INPUT:

=> d 1-40

1. 5,193,187, Mar. 9, 1993, Fast interrupt mechanism for interrupting processors in parallel in a multiprocessor system wherein processors are assigned process ID numbers; Robert E. Strout, II, et al., 395/650; 364/230.2, 281.4, 281.7, DIG.1; 395/575 [IMAGE AVAILABLE]

2. 5,187,781, Feb. 16, 1993, Shared hardware interrupt circuit for personal computers; Chester A. Heath, 395/325; 364/927, 927.93, 941, DIG.2 [IMAGE AVAILABLE]

3. 5,185,864, Feb. 9, 1993, Interrupt handling for a computing system with logical devices and interrupt reset; Francis M. Bonevento, et al., 395/275; 364/232.1, 238.3, 280.8, DIG.1; 395/700, 725 [IMAGE AVAILABLE]

4. 5,177,747, Jan. 5, 1993, Personal computer memory bank parity error indicator; Louis B. Capps, Jr., et al., 371/51.1, 21.6 [IMAGE AVAILABLE]

5. 5,150,469, Sep. 22, 1992, System and method for processor pipeline control by selective signal deassertion; Norman P. Jouppi, 395/375; 364/228.6, 231.8, 241.6, 263.2, 931.49, 933.2, 941.7, 948.34, DIG.1 [IMAGE AVAILABLE]

6. 5,131,081, Jul. 14, 1992, System having a host independent input/output processor for controlling data transfer between a memory and a plurality of I/O controllers; Craig A. MacKenna, et al., 395/275; 364/238.3, 240.5, 241.9, 242.92, 926.9, 926.93, 927.92, 927.93, 927.94, 927.98, 927.99, 933, 933.62, 935, 935.2, 935.4, 937.1, 942.8, 946.2, 964.4, DIG.2 [IMAGE AVAILABLE]

7. 5,128,943, Jul. 7, 1992, Independent backup mode transfer and mechanism for digital control computers; Bhalchandra R. Tulpule, et al., 371/9.1; 364/927.92, 927.94, 931.4, 934, 934.3, 937, 940.81, 941, 941.1, 941.2, 943.9, 943.91, 944, 944.2, 945, 946.2, 948.1, DIG.2; 395/575 [IMAGE AVAILABLE]

8. 5,123,098, Jun. 16, 1992, Method for executing programs within expanded memory of a computer system using MS or PC DOS; Michael W. Gunning, et al., 395/400; 364/231, 234, 237.2, 240, 240.1, 245, 245.2, 245.31, 246, 246.3, 254, 254.3, 280, 970, 970.5, DIG.1; 395/700 [IMAGE AVAILABLE]

9. 5,032,982, Jul. 16, 1991, Device for timing interrupt acknowledge cycles; Monte J. Dalrymple, et al., 395/550; 364/240.9, 241.2, 242, 271.5, DIG.1 [IMAGE AVAILABLE]

10. 4,972,312, Nov. 20, 1990, Multiprocess computer and method for operating same having context switching in response to a peripheral interrupt; Johannes H. den Boef, 395/725; 364/241.5, 280.8, DIG.1 [IMAGE AVAILABLE]

11. 4,930,068, May 29, 1990, Data processor having different interrupt processing modes; Tsuyoshi Katayose, et al., 395/725; 364/230, 230.2, 231, 232.9, 238.6, 238.9, 241.2, 241.3, 242.1, 243.3, 247, 247.8, 251.3, 251.4, 259, 259.7, 261.4, 262.4, 262.8, 263.2, 280, 280.8, 941, DIG.1 [IMAGE AVAILABLE]

12. 4,905,196, Feb. 27, 1990, Method and storage device for saving the computer status during interrupt; Hubert Kirrmann, 365/200, 75 [IMAGE AVAILABLE]

13. 4,888,691, Dec. 19, 1989, Method for disk I/O transfer; Paul L. George, et al., 395/700; 364/228.4, 230, 230.2, 231.4, 231.6, 232.8, 236.2, 238.3, 239, 239.6, 239.7, 240.8, 241.2, 241.4, 242, 242.1, 242.3, 242.31, 244, 244.3, 247, 247.3, 248.1, 252.3, 252.4, 260, 260.2, 265, 267, 267.2, 280,
13:01:00 COPY AND CLEAR PAGE, PLEASE

INPUT:

284, 284.2, 942.79, DIG.1; 395/425, 575 [IMAGE AVAILABLE]

14. 4,875,483, Oct. 24, 1989, Implantable cardiac pacer with programmable antitachycardia mechanisms; William Vollmann, et al., 607/15, 16 [IMAGE AVAILABLE]

15. 4,870,566, Sep. 26, 1989, Scannerless message concentrator and communications multiplexer; Ronald J. Cooper, et al., 395/325; 364/221, 221.7, 222.2, 228.4, 232.8, 238, 238.5, 240, 240.2, 240.8, 240.9, 242.1, 242.3, 242.31, 242.6, 242.91, 242.92, 247, 247.2, 247.3, 247.4, 247.5, 247.7, 247.8, 254, 254.5, 263.2, 265, 265.1, 267, 267.4, 267.7, 267.8, 284, 284.3, DIG.1; 395/425 [IMAGE AVAILABLE]

16. 4,843,890, Jul. 4, 1989, Coriolis mass flow rate meter having an absolute frequency output; Allan L. Samson, et al., 73/861.38; 364/510 [IMAGE AVAILABLE]

17. 4,736,193, Apr. 5, 1988, Programmable fluid detector; Laurence S. Slocum, et al., 340/522; 73/40.5R, 49.2, 61.43; 340/603, 604, 605, 620 [IMAGE AVAILABLE]

18. 4,727,480, Feb. 23, 1988, Emulation of a data processing system; Loren O. Albright, et al., 395/500; 364/231.4, 231.5, 241.9, 247, 280, 280.8, 280.9, DIG.1; 395/725 [IMAGE AVAILABLE]

19. 4,726,380, Feb. 23, 1988, Implantable cardiac pacer with discontinuous microprocessor, programmable antitachycardia mechanisms and patient data telemetry; William Vollmann, et al., 607/15, 16, 24, 30 [IMAGE AVAILABLE]

20. 4,719,922, Jan. 19, 1988, Stimulator apparatus; Ante L. Padjen, et al., 607/62; 128/908; 607/45, 70 [IMAGE AVAILABLE]

21. 4,710,928, Dec. 1, 1987, Method and apparatus for detecting the uncontrollable operation of a control system; Akihisa Ueda, 371/16.1, 29.1, 62 [IMAGE AVAILABLE]

22. 4,648,029, Mar. 3, 1987, Multiplexed interrupt/DMA request arbitration apparatus and method; Ronald J. Cooper, et al., 395/325; 364/222.2, 232.8, 238, 238.3, 238.5, 239, 239.6, 239.7, 239.8, 240, 240.1, 240.8, 241, 241.2, 241.5, 241.6, 241.7, 242.3, 242.31, 242.32, 242.6, 242.7, 242.92, 259, 259.2, 260, 265, 265.3, 265.6, 266.3, 266.6, DIG.1; 395/725 [IMAGE AVAILABLE]

23. 4,627,054, Dec. 2, 1986, Multiprocessor array error detection and recovery apparatus; Ronald J. Cooper, et al., 371/11.3 [IMAGE AVAILABLE]

24. 4,589,066, May 13, 1986, Fault tolerant, frame synchronization for multiple processor systems; Jack F. Lam, et al., 395/550; 364/228.3, 229, 229.2, 229.4, 230, 230.4, 239, 239.6, 240, 240.2, 240.5, 242, 242.5, 265, 266.1, 267, 267.4, 267.7, 268, 268.9, 269, 269.2, 270, 270.1, 271, 271.2, 285, 285.3, DIG.1; 371/9.1, 61, 62; 395/575 [IMAGE AVAILABLE]

25. 4,575,817, Mar. 11, 1986, Switching of programming routine supporting storage stacks; Wade H. Allen, et al., 395/275; 364/921.8, 926.9, 930, 940, 941, 943.9, 965, 965.4, DIG.2 [IMAGE AVAILABLE]

26. 4,561,442, Dec. 31, 1985, Implantable cardiac pacer with discontinuous microprocessor programmable antitachycardia mechanisms and patient data telemetry; William Vollmann, et al., 607/27 [IMAGE AVAILABLE]

27. 4,538,138, Aug. 27, 1985, Integrated security system having a
13:01:12 COPY AND CLEAR PAGE, PLEASE

INPUT:

multiprogrammed controller; Roy L. Harvey, et al., 340/521, 505, 506, 518, 523, 531, 825.06; 379/33, 49 [IMAGE AVAILABLE]

28. 4,495,571, Jan. 22, 1985, Data processing system having synchronous bus wait/retry cycle; Theodore R. Staplin, Jr., et al., 395/325; 364/222.2, 232.8, 235, 236.2, 238, 238.3, 240, 240.1, 240.2, 240.8, 241.2, 241.9, 242.3, 242.31, 242.5, 243, 243.1, 244, 244.6, 246.91, 248.1, 260, 260.1, 262.4, 262.8, 265, 265.1, 273.4, 280, DIG.1 [IMAGE AVAILABLE]

29. 4,491,904, Jan. 1, 1985, Chopper control apparatus; Michimasa Horiuchi, et al., 364/130, 184, 400; 388/811, 907.5, 909, 920, 921 [IMAGE AVAILABLE]

30. 4,484,271, Nov. 20, 1984, Microprogrammed system having hardware interrupt apparatus; Ming T. Miu, et al., 395/375; 364/229, 229.2, 232.8, 238.3, 240, 240.2, 241.2, 241.6, 242.3, 242.31, 243, 243.7, 244, 244.6, 252.3, 252.5, 259, 259.2, 261.3, 262.4, 262.8, 280, 280.2, DIG.1; 395/725 [IMAGE AVAILABLE]

31. 4,475,047, Oct. 2, 1984, Uninterruptible power supplies; Harry K. Ebert, Jr., 307/66, 87, 129; 364/492, 707 [IMAGE AVAILABLE]

32. 4,400,624, Aug. 23, 1983, Uninterruptible power supplies; Harry K. Ebert, Jr., 307/43, 66 [IMAGE AVAILABLE]

33. 4,383,295, May 10, 1983, Data processing system having data entry backspace character apparatus; Robert C. Miller, et al., 395/275; 364/229, 229.2, 234, 237.6, 238, 238.3, 239, 239.4, 239.6, 240, 240.1, 240.2, 242.1, 242.3, 242.31, 243, 243.3, 243.5, 244, 244.3, 251, 251.3, 254, 254.4, 255.4, 259, 259.4, 260, 260.1, 261.3, 262.4, 262.6, 264, 264.2, 265, 265.5, 266.5, 271, 273.4, 280, 280.8, 284, 284.2, DIG.1; 395/250, 725 [IMAGE AVAILABLE]

34. 4,352,157, Sep. 28, 1982, Data-processing apparatus having improved interrupt handling processor; Keiji Namimoto, et al., 395/725; 364/232.8, 240.1, 241.2, 241.3, 241.5, 243, 243.1, 244, 244.6, 246, 246.3, 247, 247.2, 247.6, 247.7, 252, 254, 254.3, 258, 258.1, 259, 259.1, 259.5, 259.7, 259.8, 260.4, 260.6, 261.3, 431.11, DIG.1 [IMAGE AVAILABLE]

35. 4,344,133, Aug. 10, 1982, Method for synchronizing hardware and software; William C. Bruce, Jr., et al., 395/775; 364/228.3, 232.8, 240.1, 244, 244.3, 258, 258.2, 259, 259.7, 263.2, 271, 271.1, 271.4, DIG.1 [IMAGE AVAILABLE]

36. 4,334,308, Jun. 8, 1982, Test facility for error diagnosis in multi-computer systems, particularly in multi-micro-computer systems; Hans Thinschmidt, et al., 371/29.1; 364/921.8, 927.2, 927.81, 928, 928.2, 928.4, 931.4, 939, 939.6, 940, 940.1, 940.2, 941, 943.9, 944.92, 945.5, 946.2, 946.6, 947, 947.1, 947.2, 960, 960.2, DIG.2; 371/17, 67.1 [IMAGE AVAILABLE]

37. 4,326,247, Apr. 20, 1982, Architecture for data processor; George P. Chamberlin, 395/800; 364/231.4, 231.7, 232.8, 232.9, 238.6, 238.7, 239, 239.4, 239.7, 239.9, 240, 240.1, 240.2, 242, 243, 243.2, 244, 244.3, 244.6, 247, 247.3, 247.4, 247.6, 258, 258.2, 258.3, 259, 259.2, 259.5, 259.7, 259.9, 261.3, 261.4, 264, 264.6, 271.6, 271.8, DIG.1 [IMAGE AVAILABLE]

38. 4,317,169, Feb. 23, 1982, Data processing system having centralized memory refresh; William Panepinto, Jr., et al., 395/425; 364/232.8, 234, 238.3, 240, 240.1, 240.2, 241.2, 241.3, 241.9, 242, 242.3, 243, 243.1, 246.91, 249, 249.2, 249.3, 259, 259.7, 260, 260.2, 260.4, 260.8, 261.3, 261.4, 261.5, 261.6, 262.4, 262.8, 263, 263.2, 264, 264.1, 264.5, 264.6, 265, 271, 271.4, 271.6, 271.8, 273.1, 273.4, DIG.1; 365/222 [IMAGE AVAILABLE]

13:01:26 COPY AND CLEAR PAGE, PLEASE

INPUT:

US PAT NO: 4,589,066 [IMAGE AVAILABLE] L4: 8 of 14

BSUM(7)

The synchronizer hardware produces a control signal in the form of a processor Interrupt signal at the end of each minor frame to initiate the supervisory software routine. The Interrupt signal is produced if two or more of the four (4) pulses arrive within the "time window", regardless if the other pulses arrive earlier or later than the majority. The supervisory software routine identifies any sync pulse failures--the processor associated with the failed sync pulse--and substitutes processors if the sync failure indication is less than three.

DETDESC:

DETD(19)

The selected sync pulses are applied to a local reset generator 43. Generator 43 includes sync pulse decision logic hardware which determines whether the sync pulses arrive during the "time window" and stores that information. The local sync pulse decision logic actuates a pulse generator to produce the processor Interrupt pulse. This initiates the software supervisory routine from the processor which determines the synchronizer status to see if there has been a failure, identifies any failed pulse and processor, and removes the failed sync pulse and processor.

10. 4,344,133, Aug. 10, 1982, Method for synchronizing hardware and software; William C. Bruce, Jr., et al., 395/775; 364/228.3, 232.8, 240.1, 244, 244.3, 258, 258.2, 259, 259.7, 263.2, 271, 271.1, 271.4, DIG.1 [IMAGE AVAILABLE]

US PAT NO: 4,344,133 [IMAGE AVAILABLE] L4: 10 of 14

ABSTRACT:
A digital processor capable of responding to a sync instruction for high-speed synchronization of hardware and software is provided. The sync instruction places the procesor in a stopped state and lets the processor start up again only upon receipt of an interrupt. If the interrupt is disabled by being masked, the stopped state is simply cleared and the sequencing of instructions continues without vectoring to the interrupt service routine. However if the interrupt is not disabled, the processor will handle the interrupt just as it would if it were not in the stopped state. Upon return from the interrupt service routine, the stopped state is cleared and the sequencing of instructions continues. In this way, the sync instruction provides a mechanism for synchronizing software with hardware external to the processor without the delays associated with interrupts or busy-wait loops.

DETDESC:

DETD(30)

The sync instruction provides for high-speed synchronization of hardware and software. It stops the processor and lets it start up again only when one of the interrupt lines is pulled low which indicates an interrupt signal. In this way, the instruction provides a mechanism for synchronizing software with hardware external to the processor without the delays associated with interrupts or busy-wait loops. It should be noted that the sync instruction does not cause the processor to stack any of the programmable registers. Therefore time is not wasted stacking registers when
13:55:48 COPY AND CLEAR PAGE, PLEASE

INPUT:

US PAT NO: 4,344,133 [IMAGE AVAILABLE] L4: 10 of 14

DETD(30)
it is not desired to stack the registers. The present invention allows the processor to continue from a stopped state when a masked interrupt is received. The non-maskable interrupt, NMI, will be serviced by the processor even if it is in a syncing state and in most cases will only be used in response to an emergency condition.

11. 4,326,247, Apr. 20, 1982, Architecture for data processor; George P. Chamberlin, 395/800; 364/231.4, 231.7, 232.8, 232.9, 238.6, 238.7, 239, 239.4, 239.7, 239.9, 240, 240.1, 240.2, 242, 243, 243.2, 244, 244.3, 244.6, 247, 247.3, 247.4, 247.6, 258, 258.2, 258.3, 259, 259.2, 259.5, 259.7, 259.9, 261.3, 261.4, 264, 264.6, 271.6, 271.8, DIG.1 [IMAGE AVAILABLE]

US PAT NO: 4,326,247 [IMAGE AVAILABLE] L4: 11 of 14

ABSTRACT:
A data processor having an internal address bus and a separate internal data bus which are selectively coupled to an external memory bus. The external memory bus is time shared so that it can carry memory addresses as well as data. A command shift register, at least one capture register, a timer register, a compare register, a control register, and a status register are all coupled to the internal data bus. The command shift register is capable of serially shifting data, upon command, to an output terminal. The at least one capture register is capable of being loaded from the timer register whenever a transition occurs on a predetermined input to the data processor thereby capturing the time at which the transition occurred. The compare register is used to store a digital signal equivalent to some desired time. The compare register is continuously compared for equality with the timer register and provides a signal when equality exists. The control register is capable of providing software control of preselected registers within the data processor and the status register is used to temporarily store data indicating causes of interrupts.

DETDESC:

DETD(18)

I/O status register 62 is an eight-bit register which can be read from or written into by software control and is coupled to data bus 52. Status register 62 is coupled to and receives inputs from inputs RT1, RT2, RT3, equality detector 57, and timer register 56. Status register 62 indicates the causes of interrupts and permits direct reading of the three real time input lines RT1, RT2, and RT3. The level appearing at input RT1 will be reflected by bit two of status register 62. If bit two is a logic level "0" it will indicate that the input at input RT1 is low, and if bit two is a logic level high it will indicate that the input at input RT1 is a high. In a corresponding manner, bit one of status register 62 reflects the input appearing at input RT2, and bit zero indicates the input at input RT3. Bits three through seven are set when an interrupt is detected by the input/output circuitry of processor 10. Bit three is set by a transition on input RT3, bit four is set by a transition on input RT2, bit five is set by a transition on input RT1, bit six is set when timer register 56 overflows, and bit seven is set when timer compare occurs as indicated by equality detector 57. If any one of the bits three through seven is a logic "1" and the corresponding bit in control register 47 is a logic "1", an interrupt will occur. Input RT3 can only cause an interrupt when it is in the input mode. It will be noted that the bits in status register 62 will be set to a logic "1" when the specified condition occurs regardless of the state of the
13:56:09 COPY AND CLEAR PAGE, PLEASE

INPUT:

US PAT NO: 4,326,247 [IMAGE AVAILABLE] L4: 11 of 14

DETD(18)
interrupt enable bits in control register 47, however, interrupts will only be generated when the corresponding enable bit in register 47 is a logic one. The bit in status register 62 which causes the interrupt will be cleared to a logic "0" by the hardware when the interrupt is recognized. Also, the status bit or bits may also be cleared by software.

DETDESC:

DETD(20)

The last software instruction within each of the interrupt handling routines stored in foreground software is a return from interrupt RTI instruction. If no interrupts are active when the interrupt handling routine finishes servicing the last interrupt, the execution of the return from interrupt RTI instruction causes program control to be returned to the background memory program. If an interrupt condition still exists when the RTI is executed, another interrupt will occur immediately with the appropriate interrupt vector location being used because the effect of the RTI is the same as executing a jump-to-subroutine (JSR) instruction and a new vector address is provided for fetching the jump address to be executed by the JSR instruction. Bits three through seven of status register 62 may be written by software thereby causing an interrupt if the interrupt is enabled by the associated bit in control register 47. Bits zero through two of status register 62 cannot be written by software. Only ten bits are fetched from memory for an interrupt vector when an interrupt occurs. The three high order bits AD10 through AD12 are hardware generated.

=>

INPUT:

=> d 2,8,10,11 cit hit

2. 5,128,943, Jul. 7, 1992, Independent backup mode transfer and mechanism for digital control computers; Bhalchandra R. Tulpule, et al., 371/9.1; 364/927.92, 927.94, 931.4, 934, 934.3, 937, 940.81, 941, 941.1, 941.2, 943.9, 943.91, 944, 944.2, 945, 946.2, 948.1, DIG.2; 395/575 [IMAGE AVAILABLE]

US PAT NO: 5,128,943 [IMAGE AVAILABLE] L4: 2 of 14

ABSTRACT:
An interrupt is provided to a signal processor having a non-maskable interrupt input, in response to the detection of a request for transfer to backup software. The signal processor provides a transfer signal to a transfer mechanism only after completion of the present machine cycle. Transfer to the backup software is initiated by the transfer mechanism only upon reception of the transfer signal.

SUMMARY:

BSUM(15)

According to the present invention, the transfer method and mechanism, when activated, sends a non-maskable interrupt to all of the channel processor(s) when a majority of channels detect (by means of a sever request, a user request or any other mechanism) a generic software failure; each of the processors then sends an acknowledge signal in response to the non-maskable interrupt after concluding the machine cycle in which it is engaged at the time it receives the interrupt; the acknowledge signal, which is purely a hardware driven signal, is then used to transfer the signal processor's program memory from a primary program memory to a backup program memory.

8. 4,589,066, May 13, 1986, Fault tolerant, frame synchronization for multiple processor systems; Jack F. Lam, et al., 395/550; 364/228.3, 229, 229.2, 229.4, 230, 230.4, 239, 239.6, 240, 240.2, 240.5, 242, 242.5, 265, 266.1, 267, 267.4, 267.7, 268, 268.9, 269, 269.2, 270, 270.1, 271, 271.2, 285, 285.3, DIG.1; 371/9.1, 61, 62; 395/575 [IMAGE AVAILABLE]

US PAT NO: 4,589,066 [IMAGE AVAILABLE] L4: 8 of 14

ABSTRACT:
The processors in a redundant, multiprocessor system are synchronized at the minor frame level with a combination of hardware and a local software supervisory routine. Each processor includes an interface synchronizer unit which receives synchronizing pulses from a selected number of the processor synchronizer units at the end of each minor frame. Sync decision logic circuits in each synchronizer determine whether the synchronizing pulses arrive within a predetermined time period or "window" (2 usecs, for example) indicating synchronization between the processors. A control, processor "Interrupt", signal is generated whenever a majority (>2) of the four (4) sync signals are received at the end of the minor frame. The local processor then initiates the supervisory software routine, which checks the status of the synchronizer for failure indications, isolates and records the faulty sync pulses and then replaces any faulty processor with another processor.

SUMMARY:
13:55:31 COPY AND CLEAR PAGE, PLEASE

INPUT:

US PAT NO: 5,179,368 [IMAGE AVAILABLE] L4: 4 of 51

DETD(95)
hardware interrupt system with a software interrupt of a subroutine call. While the prior art required actual determination of light pen status at each sub-program call, the invention allows that information to be determined at vertical retrace time, then passed very quickly to the calling program upon demand (i.e., through the memory areas 765, 805), further reducing the computational overhead associated with control sub-program.

US PAT NO: 5,177,747 [IMAGE AVAILABLE] L4: 5 of 51

SUMMARY:

BSUM(4)

Parity checking is a well known method for detecting errors in transmitting data. In accordance with such method, a parity bit is or is not added to a packet, e.g. a byte, of binary digits so as to maintain the total number of bits, including the parity bit, as an odd or an even sum. When the packet is transmitted, the total number of bits is counted and if the sum is not odd or even as it is supposed to be, a parity error has occurred. Current high performance personal computers have thirty two bit wide memory data paths in which data is arranged in four eight bit bytes each byte being associated with one parity bit. A parity checking circuit is connected to a data path and upon detecting a parity error, it sends a signal that latches up a flip flop which generates a parity check signal. The parity check signal in turn causes a hardware interrupt to be sent to a processor and a software interrupt handling routine analyzes the error, displays an error code on a display, and halts operation of the computer.

US PAT NO: 5,121,472 [IMAGE AVAILABLE] L4: 6 of 51

SUMMARY:

BSUM(6)

In more specific terms, within the IBM (tm) PC/PS2 personal computer hardware environment, there are two standard keyboard handling routines called "interrupt nine (INT9)" and "interrupt sixteen (INT16)". INT9 is a hardware interrupt: the system hardware causes the INT9 keyboard interrupt handler to be executed every time that the user physically presses or releases any key on the keyboard. Upon execution of INT9, the standard keyboard handler routine reads a number (called the "scan code") of the activated key using an IN instruction to fetch the number from I/O Port 60. The scan code is then usually converted into a character code (e.g. the character "a") and is stored in a buffer or memory location called the keyboard buffer. INT16 is a software interrupt routine which is executed only when it is called by a program, such as Lotus 1-2-3 for example, when the program is ready for an input value from the keyboard. The standard INT16 handler obtains a character from the keyboard buffer and passes it along to the program requiring the character. Macro generators typically include their own keyboard interrupt handlers for INT9 and/or INT16.

US PAT NO: 5,060,151 [IMAGE AVAILABLE] L4: 7 of 51

DETDESC:
13:15:08 COPY AND CLEAR PAGE, PLEASE

INPUT:

US PAT NO: 5,060,151 [IMAGE AVAILABLE] L4: 7 of 51

DETD(34)

The hardware interrupt from the speed sensor takes precedence over the software interrupt and calls the motor driver routine 190 to control the speed of the apparatus. After execution of the motor driver routine 190, the microprocessor 100 will either return to the main monitor loop or service the software interrupt, if it is waiting.

DETDESC:

DETD(35)

Therefore, the execution sequence of the program is once through the initializing routine upon startup or reset and then to the main monitor routine for constant execution. The execution of the monitor routines are interrupted by the software interrupt every 256 times per second, and by the hardware interrupt at times depending upon the speed of the motor. After the interrupts have been serviced the program returns to the execution of the main monitor until stopped.

US PAT NO: 5,018,114 [IMAGE AVAILABLE] L4: 8 of 51

DETDESC:

DETD(83)

Referring to both FIGS. 9 and 10 initialization of data acquisition system 214 causes a sampling. These samples go into a transmit DMA buffer 252. When transmit DMA buffer 252 is filled, DMA controller 222 causes a hardware interrupt 224. Hardware interrupt 224 in turn causes DAS transmit interrupt routine 250 to execute by way of the mode 230 and interrupt vector 228 path. During execution of the transmit interrupt routine, a software interrupt 254 is generated to cause execution of a device driver routine within a device driver 256. Device driver 256 causes a switching of the DMA channels within DAS 214 and reprograms DMA controller 222 for the next transmit DMA buffer 252. Continuing within the data acquisition system transmit interrupt routine 250, code word data contained within transmit user buffer 246 will be passed to data acquisition system 214 one code word per hardware interrupt. This process continues until transmit user buffer 246 is emptied. The code words are converted into analog signals by data acquisition system 214 which controls transmit board 258. Transmit board 258 is one and the same as transmit cards 42 and 104 shown respectively in FIGS. 2 and 5. Transmit board 258 outputs an analog acoustic signal to the acoustic transducer interface system 210 to cause the propagation of acoustic tones through the aqueous medium.

DETDESC:

DETD(85)

Initialization of the receive mode 260 causes DAS receive interrupt routine 226 and interrupt vector 228 to be installed. Mode switch 230 is set to the receive mode. Upon switching mode switch 230 to receive mode, DAS 214 is initiated and samples analog signals from receive boards 212. As receive DMA buffers 220, shown in FIG. 10, become filled, DMA controller 222 generates hardware interrupt 224. Hardware interrupt 224 causes DAS receive interrupt routine 226 to execute by way of interrupt vector 228 and mode switch 230. Data acquisition system receive interrupt routine 226 generates a
13:16:36 COPY AND CLEAR PAGE, PLEASE

INPUT: